)ne-Time Verifier-Based
ıncrypted Key Exchange

Michel Abdalla
ENS – France

Olivier Chevassut
LBNL – DOE - USA

David Pointcheval
CNRS-ENS – France

PKC '05
Les Diablerets, Switzerland
January 24th 2005

;ummary

- *Authenticated Key Exchange*
- *Password-Based Authentication*
  - *EKE and OKE*
  - *Security Results*
- *Enhanced Security against Corruption*

Authenticated Key Exchange

Two parties (Alice and Bob) agree on a ***common*** secret key sk, in order to establish a secret channel

- Basic security notion: ***semantic security***
  - only the intended partners can compute the session key sk
- Formally:
  - the session key sk is indistinguishable from a random string r, to anybody else

Further Properties

- ***Mutual authentication***
  - They are both sure to **actually** share the secret with the people they think they do
- ***Forward-secrecy***
  - Even if a long-term secret data is corrupted, previously shared secrets are **still** protected

›assive/Active Adversaries

- ***Passive adversary***: ***history*** built using
  - the **execute**-queries $\rightarrow$ transcripts
  - the **reveal**-queries $\rightarrow$ session keys
- ***Active adversary***: *entire control of the network*
  - the **send**-queries

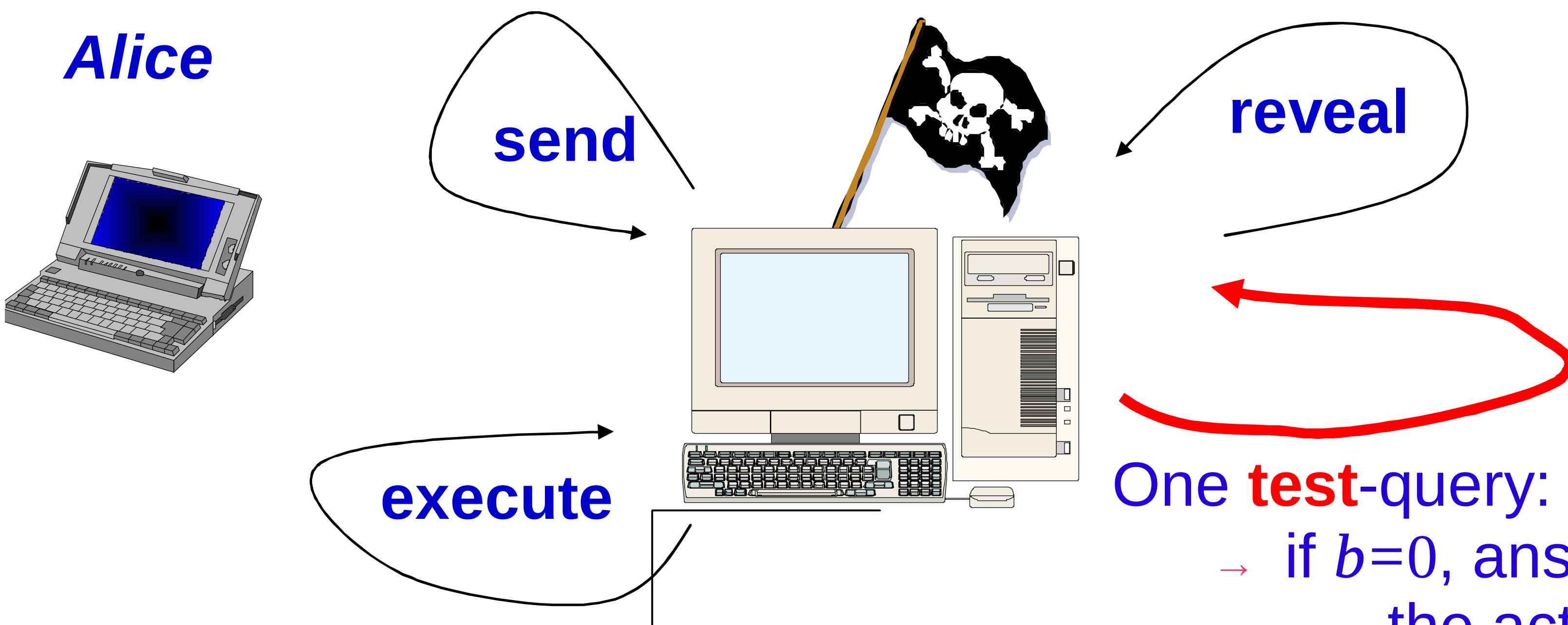
    active, adaptive adversary on concurrent executions

    → to send message to Alice or Bob
    (in place of Bob or Alice respectively)

    → to intercept, forward and/or modify messages

emantic Security

As many **execute**, **send** and **reveal** queries as the adversary wants



0/1
try to guess b

One **test**-query: flip a coin b,
→ if b=0, answer the actual secret data
→ if b=1, answer a random string r

·reshness

A_i and B_j: two instances of Alice and Bob

- the adversary asks a **reveal** to A_i
- the adversary asks the **test** to B_j

Freshness:

- the instance has accepted (holds a key!)
- neither the instance nor its partner has been asked for a **reveal** query

Foward Secrecy: Corrupt-Query

***Forward Secrecy**: corruption of long term keys*

- *the **corrupt**-queries ® long-term key*

FS-Freshness:

- the instance has accepted (holds a key!)
- neither the instance nor its partner has been asked for a **reveal** query
- (neither the instance) nor its partner has been asked for a **corrupt** query

$\Rightarrow$ Diffie-Hellman provides the Forward Secrecy

)iffie-Hellman Key Exchange

G = $<g>$, cyclic group of prime order q

- Alice chooses a random $x \in \mathbf{Z}_q$,
  computes and sends $X=g^x$
- Bob chooses a random $y \in \mathbf{Z}_q$,
  computes and sends $Y=g^y$
- They can both compute the value

$$K = Y^x = X^y$$

ʼroperties

- Without any authentication, no security is possible:
  man-in-the-middle attack

$\Rightarrow$ some authentication is required

- If flows are **authenticated** (MAC or Signature), it provides the forward secrecy under the **DDH Problem**
- If one derives the session key as $sk = \mathrm{H}(K, \ldots)$, in the random oracle model, the forward secrecy is relative to the **CDH Problem**

Password-based Authentication

Password (short – low-entropy secret – say 20 bits)

- exhaustive search is possible
- basic attack: **on-line exhaustive search**
  - the adversary guesses a password
  - tries to play the protocol with this guess
  - failure $\Rightarrow$ it erases the password from the list
  - and restarts…

after 2^{20} attempts, the adversary wins

)ictionary Attack

The on-line exhaustive search

- *cannot be prevented*
- *can be made less serious (delay, limitations, …,*

'e want it to be the best attack…

The off-line exhaustive search

- *a few passive or active attacks*
- *transcripts Þ password, by an off-line check*

this is called dictionary attack

Þ our GOAL: prevent dictionary attacks

ːxample: EKE

ıe most famous scheme: Encrypted Key Exchange

ither one or two flows are encrypted with the passworc

Alice | Password π | Bob

$\mathbf{Z}_q$, $X=g^x$ | Alice, $X' = \mathbf{E}_\pi(X)$ $\longrightarrow$

$X = \mathbf{D}_\pi(X')$
$y \in \mathbf{Z}_q$, $Y=g^y$
$K=X^y$

$\longleftarrow$ Bob, $Y' = \mathbf{E}_\pi(Y)$

= $\mathbf{D}_\pi(Y')$
$K=Y^x$

$sk=H(\text{Alice, Bob}, X', Y', \pi, K)$

Alice | Password π | Bob

$x \in \mathbf{Z}_q$, $X=g^x$ | Alice, X $\longrightarrow$

$\longleftarrow$ Bob, $Y' = \mathbf{E}_\pi(Y)$ | $y \in \mathbf{Z}_q$, $Y=\varsigma$

$Y = \mathbf{D}_\pi(Y')$, $K=Y^x$
$k_1=H_1(\text{Alice, Bob}, K)$ | k_1 $\longrightarrow$ | $K=X^y$
k_1 correct ˀ

$sk=H(\text{Alice, Bob}, X', Y', \pi, K)$

$\mathbf{E}_\pi$: ideal cipher

$\mathbf{E}_\pi(X) = \mathrm{H}(\pi).X$ in ROM

EKE - OKE

OKE: Open Key Exchange

- first flow sent in clear (open)
- forward secrecy = CDH [MKe02: PAK]

EKE: Encrypted Key Exchange

- both flows encrypted
- semantic security = CDH

KE: Forward secrecy
= open problem
[MKe02: PAK]

Alice	Password π	Bob
$x \in \mathbf{Z}_q$, $X = g^x$	Alice, $X' = H(\pi).X$ $\longrightarrow$	
		$X = X' / H(\pi$
$Y = Y' / H(\pi)$	$\longleftarrow$ Bob, $Y' = H(\pi).Y$	$y \in \mathbf{Z}_q$, $Y = g$
$K = Y^x$		$K = X^y$
	$sk = H(\text{Alice}, \text{Bob}, X', Y', \pi, K)$	

Reasons...

Proof of semantic security:

- sequence of indistinguishable games, such that at the end the simulation does not use the password

Þ the password can be chosen at the very end to check whether or not the adversary had won

In the forward-secrecy game:

- the password has to be chosen when the corrupt query is asked, and then the adversary knows the password
- he can ask reveal or hash queries on previous keys (when the password was unknown to the simulator)

Þ consistency?... Decisional Oracle... Þ Gap Problem

KE: Security Results

Assumptions

- **two different masks with H_1 and H_2**
- **random-oracle model for H, H_1, and H_2**

Alice	Password π	Bo
$x \in \mathbf{Z}_q$, $X = g^x$	Alice, $X' = H_1(\pi).X$ →	$X = X'$ / $H_1(\pi)$
$Y = Y'$ / $H_2(\pi)$	← Bob, $Y' = H_2(\pi).Y$	$y \in \mathbf{Z}_q$, $Y=$
$K = Y^x$	$sk = H(\text{Alice}, \text{Bob}, X', Y', \pi, K)$	$K = X^y$

Semantic security of EKE :

advantage $\leq 2\, q_s/N + 3q_h^2\, Succ^{CDH}(t') + e$

Forward Secrecy of EKE :

advantage $\leq 2\, q_s/N + 4\, Succ^{GDH}(t', q_h) + e$

$Succ^{GDH}(t,q)$ **= Probability to solve the CDH problem, within time t, after q calls to a DDH oracle**

nproved Security

- ***Protecting against server corruptions: verifier-based authentication***
  - ***Alice knows a password*** p***,***
  - ***Bob just knows a verifier of the password*** $v = f(p)$***,***
    - → v ***is the actual password,***
    - → ***then Alice proves her knowledge of*** $p = f^{-1}(v)$***, in ZK***

nproved Security (Con'd)

- ***Protecting against client corruptions: one-time password authentication***
  - ***the actual password is $v_n = f^n(p)$***
  - ***at the end the client sends, encrypted under the new session key, $v_{n-1} = f^{n-1}(p)$, which validity can be easily checked***
  - ***the next password will be v_{n-1}***